060611/Thomas Munther
Halmstad University
School of Information Science, Computer and Electrical Engineering

Laboratory 5

Length about 2 hours with supervision.
Even if you completed the whole Lab-PM during these 2 hours you should try to find the time to work through this Lab-PM thoroughly afterward.
The last page contains hand-in assignments. You should try to solve them within a week to be able to follow the pace in the course.
Everything that follows after >> is written in the Matlab Command Window and indicates what you as a user should write. The % is intended as a comment to that specific command.

This laboratory will deal with linear equations, the least square method, condition numbers over- and under-determined equations and of course other concepts as well.

Example 1: We have an equation system with three unknown variables and three equations.

3w-2y+4z=8
5w+8y-6z=-5
9w-2y+7z=-17

How will we use Matlab to find solutions for w, y and z?
First, we need to think about this problem in its matrix notation which is:

AX=b

where X is a column vector containing the unknown (w, y and z), A and b are:

```
b=          A=
   8            3  -2  4
  -5            5   8  -6
 -17            9  -2   7
```

This topic is studied in linear algebra. See Wikipedia and online tutorials if this is new to you.

Once we define vector b, and matrix A in Matlab, we can find exact solutions by:

```
>>A2 = inv(A);
>> X= A2*b
```

Or, more efficiently as:

```
>> X=A\b
```

In our case X becomes:
X=
-36.7778
71.2778
65.2222

Also try a specific command:
>> lsqr(A,b)

What is the answer?

The solution for X is not 100% accurate, due to rounding errors and limitations of the calculations. The condition number can be examined which tells us how reliable the solution will be. The condition number is always ≥ 1. The greater the sensitivity, the greater the number, the less reliable the solution. Calculate the sensitivity in our system.

>> cond(A)

As you have seen there are many different commands to use for solving equation systems. I will mention a few. They use the same argument as lsqr. Try the following: *pcg, qmr, symmlq, minres* . Some of them are successful and one or two fail.

Which one of these methods is correct ?

In theory, $AA^{-1} = I$ (the identity matrix). However, for a badly conditioned matrix A things are not so simple.
Lets try with the Matrix A we already have:

>>A*inv(A)

This should give you close to the identity matrix (in matlab you get the 3x3 identity matrix by "eye(3)"). Even though it is close, its not exact.

>>A*inv(A)- eye(3)

To measure how the elements are different from the expected, we can take the magnitude of each element of the above comparison, and sum them:

>> sum(sum(abs(A*inv(A) - eye(3))))

Return to the matrix A and change the element A(3,3). See below !

A =
3.0000 -2.0000 4.0000
5.0000 8.0000 -6.0000
9.0000 -2.0000 7.5600

What happens to the condition number? Will AA^{-1} be closer or further

from the identity matrix?

Underdetermined systems:

The equation system we discussed had a square matrix A (nxn). What happens if we do not have the same number of equations as unknown variables ? See Example 1 !

If we have fewer equations than unknowns, then we have a under-determined system. A is then an mxn matrix, where m<n. It means that we have infinitely many solutions, but Matlab only presents one of these solutions without any warning of all the others.
Solve the equation system below:

$x + y + z = 2$
$x + 2y + 3z = 3$

Once again, use:
>>A\b

What solution is presented ?

Overdetermined systems:

If we have more equations than unknowns, then the system is over-determined (there is probably no exact solution). A is then an mxn matrix, where m>n. This is a very realistic case in real life.

Suppose we have 58 measurements of current and voltage over a resistor, and we wish to determine the resistance using Ohm´s law. In theory these measurements would lie on a straight line on a U-I plot, but in practice this is never exactly the case. The best possible line, will never go exactly through all the points, due to inaccuracy in both measurement, as well as representation on the computer. The best possible line is the one where the distance of each point is small. More specifically, the sum of all squared distances should be minimum according to the least square method. Once again, we use:
>>A\b
In this case.

Solve the following example where we have 3 equations, but only 2 unknowns.

$x + y = 2$
$x + 2y = 0$
$x + 3y = -1$

Our coefficient matrix is A and the measurement vector is b: AX=b

$$A = \begin{matrix} 1 & 1 \\ 1 & 2 \\ 1 & 3 \end{matrix} \qquad b = \begin{matrix} 2 \\ 0 \\ -1 \end{matrix}$$

What do x and y become according to the least square method ?

In the Example 2 below we solve an overdetermined equation system without any

knowledge about more advanced commands. Instead we rely on our mathematical background. First specify the nomal equations !

Example 2: We have the following output from an experiment: y=[0 1.1 1.93 3.3] and the corresponding input is x=[0 1 2 3]. Let us now look for a function that adapts as good as possible to the data set. Try to find a model that suits our data. First we make a plot of the data! See Figure 1a !
In all real-life measurements we always have the presence of disturbances and noise. These will of course have some impact on our measurement and give us some error. It will happen no matter how good we are in modeling. We will not find a perfect match between our model and the measured data.

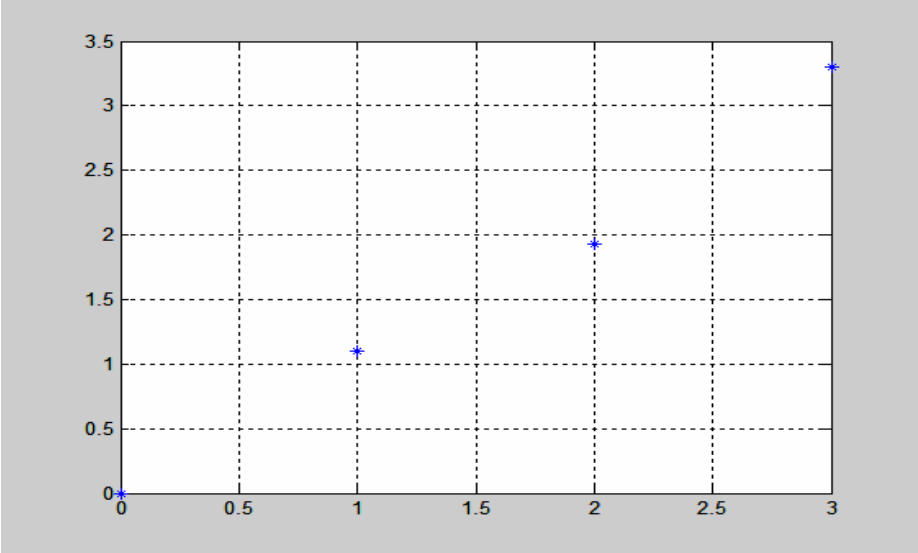

Figure 1a

In Figure 1a it seems that a suitable function could be a straight line.
$y=a_1x + a_2$, where a_1 and a_2 must be determined. If we use our y- and x-values we get the following:

$3.3 = a_1 * 3 + a_2$
$1.93 = a_1 * 2 + a_2$
$1.1 = a_1 * 1 + a_2$
$0 = a_1 * 0 + a_2$

We now have 4 equations and 2 unknown. Now let us look for a best possible solution (least square) that fits our data, where the sum $(a_1x_i + a_2 - y_i)^2$ should be as small as possible. Index i goes from i=0 to 3.

We rewrite the 4 equations above to a matrix equation Y=X*a,
where Y= 3.3 and X= 3 1
1.93 2 1
1.1 1 1
0 0 1

and $a = \begin{matrix} a_1 \\ a_2 \end{matrix}$.

We need to solve the vector a from the matrix equation, but in order to do so X must be inverted, which can be achieved only if X is a square matrix.
The matrix equation is written like:
$X^T*Y = X^T*X*a$

X^T*X is a square matrix, and we will try to invert it. If the inverse exists:
$a = (X^T*X)^{-1}* X^T*Y$
the vector a that we get contains the values a_1 and a_2. This is the best solution according to the least square method.
It is time to try the solution.
Write the following in Matlab:
>> a=inv(X'*X)*X'*Y

a= if we plot this, it will give a straight line. See figure 1b.
1.0730
-0.0270

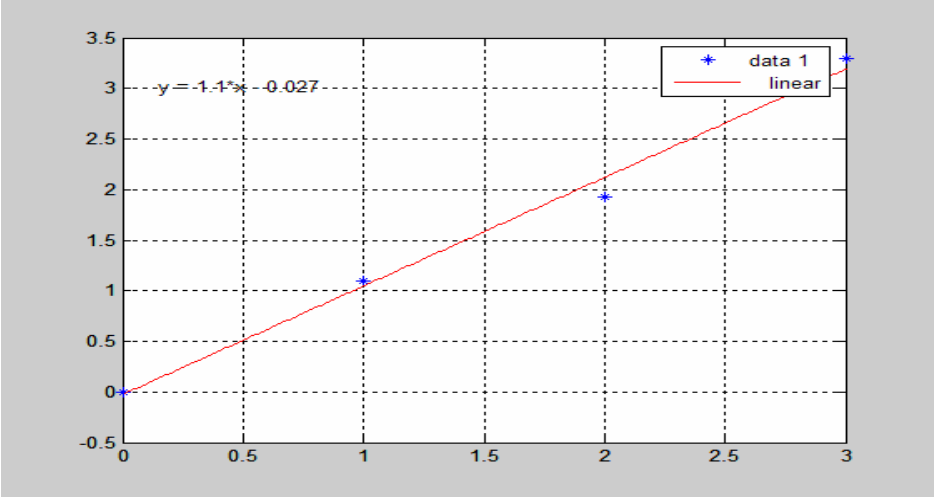

Figure 1b

It is not necessary always to adapt the data to a linear function, but in this example the data seemed to fit nicely for a first order function.
For other data we might have looked for a quadratic, cubic or a fourth-order polynomial.

$y=a_1x^2 + a_2x + a_3$, quadratic function

$y=a_1x^3 + a_2x^2 + a_3x + a_4$, cubical function

Eigenvalues and eigenvectors.

Eigenvalue problems are commonly first introduced to students in courses in linear algebra. Eigen value problems reappear in virtually all technical fields.

Let A be a nxn matrix, and assume that the X vector will be separated from the zero vector (at least one of w, y and z are NOT zero).

An eigen vector to matrix A, is a vector X (non-zero) for which the following holds

$AX=\lambda X$

for some number λ. We say that the eigen vector X has eigenvalue λ.

Eigenvalues and eigenvectors can be complex. A nxn matrix has always exactly n eigenvalues.

Example 3:

Let us solve the following example. We have a matrix A=[1 2 3; 4 5 6; 7 8 7]. Let us find the eigenvectors to the matrix.

```
>> [B,D]=eig(A)

B =

   -0.2488   -0.5718    0.5956
   -0.5686   -0.3273   -0.7589
   -0.7841    0.7523    0.2634
```

D =

```
   15.0235         0         0
         0   -1.8018         0
         0         0   -0.2217
```

We can find the eigenvectors as columns in matrix B, and the eigen values along the diagonal of D. Thus, the first eigenvector is B(:,1), and it has eigenvalue D(1,1). Check that the relation

$$AX=\lambda X$$

Holds for all eigenvectors and eigenvalues returned by the "eig" function.
(Hint: what does A*B(:,1) give you?)

Systems of differential equations
In an ordinary differential equation we often have a time-dependent variable like x(t). Here we know that the time derivative or growth (velocity) $x'(t)$ depends on x(t). Sometimes we have several such differential equations linked to each other, for instance as below:

$$x_1'(t)=a_{11}x_1(t) + a_{12}x_2(t) + ...+ a_{1n}x_n(t)$$
$$x_2'(t)=a_{21}x_1(t) + a_{22}x_2(t) + ...+ a_{2n}x_n(t)$$

.
.
.
$$x_n'(t)=a_{n1}x_1(t) + a_{n2}x_2(t) + ...+ a_{nn}x_n(t)$$

This is a nxn-system. We have n first order differential equations, where a_{ij} are the coefficients.

A differential equation $x'(t)=ax(t)$ has a general solution according to single variable calculus: $x(t)=Ce^{at}$

If we have n number of differential equations in a system, we will get something similar as solution, connected to the eigenvalues of the system.

Example 4:
Consider this simple system of diff equations:
$x_1'(t)=x_1(t) + 2x_2(t)$
$x_2'(t)=2x_1(t) -2x_2(t)$

We collect all coefficients in a matrix A=[1 2; 2 -2].

Calculate the eigenvalues and normalized eigenvectors for the coefficient matrix A.
>> [X,D]=eig(A)

X =
-0.4472 -0.8944
0.8944 -0.4472

D =
-3 0
0 2

The general solution to the differential equations can be identified straight away from the X and D matrix. The solution is:

$x_1(t)= a_1e^{-3t}(-0.4472) + a_2e^{2t}(-0,8944)$ (Eq 1)
$x_2(t)= a_1e^{-3t}(0.8944) + a_2e^{2t}(-0.4472)$ (Eq 2)

The values of the constants a_1 and a_2 can be found only if we have initial values of the differential equations.

Assume initial values for $x_1(t)$ and $x_2(t)$ and that these are:
$x_1(0)=1$ and $x_2(0)=4$.
This gives us an entire new equation system if the initial values are inserted in Eq1 and Eq 2.

$-0.4472\ a_1 - 0.8944\ a_2 = 1$
$0.8944\ a_1 - 0.4472\ a_2 = 4$
Now we have an equation system with 2 unknowns and 2 equations, from which we find a_1 and a_2 in the following way:
.
X*a=b where a=[a_1 ; a_2] , b=[1 ; 4] and X=[-0.4472 -0.8944; 0.8944 -0.4472]

The solution becomes:

```
>> a=X\b
a =
    3.1305
   -2.6833
```

It is now possible to plot the solutions $x_1(t)$ and $x_2(t)$ during the first second.

```
>> t=0:0.1:1;
>> x1=a(1)*exp(D(1,1)*t)*X(1,1)+a(2)*exp(D(2,2)*t)*X(1,2);
>> x2=a(1)*exp(D(1,1)*t)*X(2,1)+a(2)*exp(D(2,2)*t)*X(2,2);

>> plot(t,x1,t,x2,'-.'),grid ,legend('x1(t),'x2(t)')
>> title('Solutions for Example 2')
```

See the result of the plot below in Figure 2 !

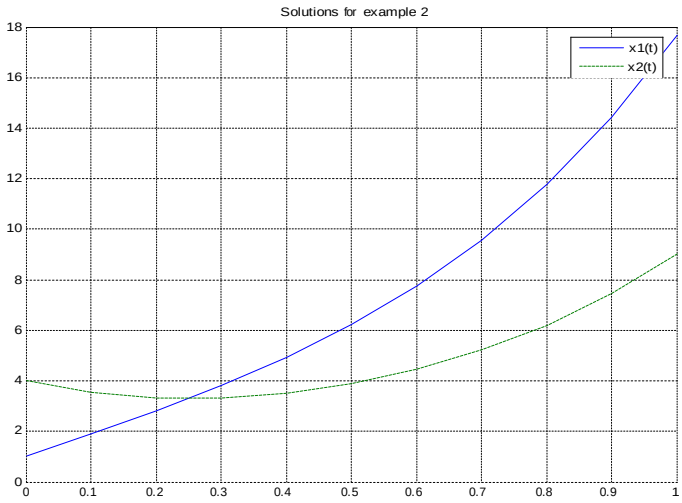

Figure 2

Sparse Matrices

A matrix is sparse when it has few nonzero elements. This can be useful to know, when we are dealing with calculations and memory allocation. In Matlab we can request that a matrix be reduced to a sparse matrix. The benefit is that only the nonzero positions and values are stored in memory. Create an identity matrix !

```
>> A=eye(1000)    % The diagonal elements are 1, the others are 0.
                  % The size of the matrix is 1000x1000.
```

Check how many bytes are allocated in the memory (Workspace)!

Now use the sparse command to save memory.
>> B=sparse(A) % A has become a sparse matrix.
>> whos
Compare the memory allocation between A and B. What do you think?
Also check the number of rows and columns in A and B !

Does it matter, when it comes to time demand ? Let us make a simple calculation.
Multiply the matrix A with the scalar 3.
>> 3*A
We can simply measure the time to execute the command. This can be done using the commands *tic and toc*.

>> tic, 3*A, toc % tic starts clock and toc stops clock.
The time elapsed can be found in the command window.
Compare it with the scalar multiplication of the sparse matrix below.

>> tic, 3*B, toc % Starts timer for the command 3*B.
We can see a clear difference, so it could matter if we must save both time and memory.

In the table below we give some examples of Matlab commands that can be applied to sparse matrices.

speye(A)	Gives a sparse matrix, with ones on the main diagonal.
sprand(A)	Gives a sparse matrix with random numbers (0-1) on the nonzero positions.
sprandn(A)	Same as above, but the elements are normally distributed (-1 to 1).
spones(A)	Gives a sparse matrix with ones as the only nonzero elements.
full(A)	Makes a full matrix from a sparse matrix A.
nnz(A)	Gives the number of nonzero elements in the matrix A.
issparse(A)	Returns 1, if the matrix is sparse or otherwise 0.
nonzeros(A)	Returns a vector with all of the nonzero elements in A.
find(A)	Gives an index for all nonzero elements in matrix A, regardless whether A is a sparse matrix or not.

We will conclude this with a somewhat more amusing exercise. The command *gplot* can be used to connect coordinates with the aid of a matrix.
I have put 21 pairs of coordinates in a matrix *cord* and described in another matrix A how they should be connected.
These matrices can be obtained from my homepage by loading the mat-file *koordinater*.
The matrix A has the size 21x21 and cord 21x2.
A indicates how the coordinates should be connected. If for instance an element A(1,2) is nonzero, actually 1, this implies that coordinate pair number 1 and number 2 should be linked. If there are any other connections between the different coordinate pairs, this results in a nonzero element in matrix A.

What use can we have from this plot of nodes? We could investigate what kind of connections we have between different places in a certain geographical region.
We will also use it for illustration later on.
>> gplot(A,cord)
>> axis([0 1.6 0 1])

Homework assignments for Laboratory 5

1. We believe that a quadratic function would be possible to adapt the following measured output data: y=[0.51 0.82 1.1 1.22 1.34 1.4] and input data x=[0 0.3 0.8 1.1 1.6 2.25]. Find the best quadratic function for: $y=a_1x^2 + a_2x + a_3$ according to the least square method.
   Make an m-file named "**e5_1.m**" that plots the quadratic function and the data in the same plot. Use the normal equations to solve the problem !

2.a) Create an m-file "**e5_2.m"** that requests 5 pairs of x and y values (5 points in the plane). You should make a plot of these with (points marked with an *) and perform a least square fit. The fit should be a linear function. The pairs should lie in the interval 0-15. Also, make sure the resulting figure has a nice title and a legend

Hint:

```
figure; axis([0 15 0 15]); hold on;

uiwait(msgbox('click 5 points in the figure to fit a curve through'));
[x, y] = ginput(5);
plot(x,y,'x');
```

2.b) In addition to what "**e5_2.m**"is already doing, make it open a new figure, and in it, plot all the points (5 in total) given by the user. Draw a connecting line from each point, to every other point. Hint: use gplot.

3. Write an m-file called "**e5_3.m"** that solves the equation system below consisting of 3 first-order differential equations.

$$x_1'(t) = x_1(t) + x_2(t) - 2x_3(t)$$
$$x_2'(t) = 2x_1(t) - 2x_3(t)$$
$$x_3'(t) = -2x_1(t) + 2x_2(t) + x_3(t)$$

Produce a plot of the solutions in the time interval 0-2 seconds.
Assume the following initial values: $x_1(0)=2$, $x_2(0)=1$ and $x_3(0)=1$

You must hand in a m-file for each problem. The m-file should begin with 2 comment lines stating when this file is created and by whom. Mail the m-files to me in a compressed format like zip.files.

ALL M-FILES should be able to run correctly on first try. Test on your system that it is working stand-alone by typing:

>>close all;clear all;
>> e5_1;